f·i·d·ĕ·

Fuzzy
Inference
Development
Environment

QUICK START GUIDE

APTRONIX
Tel: 408-428-1888
Fax: 408-428-1884
Fuzzy Net: 408-428-1883 (Setting: 8/N/1)

Part Number SGM03FID-MC

APTRONIX

▶ ▶ ▶ ▶ ▶

Chapter 1 Introduction

Welcome

Welcome to Fide, your gateway to a **Fuzzy Inference Development Environment.**

What is Fide?

Fide is a complete environment for developing fuzzy inference units and fuzzy inference systems. With Fide, engineers can:

- Model human decision making strategies by FIL (fuzzy inference language).
- Employ many debugging tools which lead the user to the bug locations in the FIL source code.
- Simulate the results before fabricating the product.
- Link Fide modules with external modules into your own application.
- Generate code for specific chips, such as Motorola MC6805.

What does Fide do?

A fuzzy inference system consists of two types of interacting units (modules): fuzzy units and nonfuzzy units. Fide is designed to cover the entire spectrum of the design process, at both the unit and system level.

Running on Microsoft Windows, Fide provides support at each stage of the development of a fuzzy inference system. You have tools to develop, test, debug, and verify your designs. Some of the tools included in Fide are the following:

- EDITOR - allows the user to create English-like statements to define various portions of a fuzzy inference system. The language is called FIL, which stands for **Fuzzy Inference Language**. For readers already familiar with fuzzy logic, the FIL allows you to build rules and membership functions, specify inference methods, defuzzification methods, and fuzzy

logical operators. Also, EDITOR allows you to edit membership functions graphically.

- COMPILER - compiles FIL source code and generates object code. The object code will be used by Debugger, Composer, and RTC.

- DEBUGGER - Fide Debugger provides three tools: Tracer, Analyzer, and Simulator.

  - Tracer - traces from output to source code. You can set input variables to test values and observe the output values. The Tracer also allows you to inspect every step in the inference process that produces the output values.

  - Analyzer - displays the three dimensional surface, contour, and side views of the input/output transfer function. You can check the function in detail and inspect the surface in a variety of perspectives. The Analyzer also allows you to trace to the source code for any point on the surface.

  - Simulator - simulates the dynamic behavior of the unit for a set of test data. The Simulator also allows you to trace to source code for any value.

- COMPOSER - allows the user to create a complete system that combines both fuzzy modules and non-fuzzy modules using the FCL (Fide Composer Language). The Composer provides a graphic editor to design fuzzy inference systems. It also has system level debugging tools, such as data flow and system simulation. The Composer allows you to trace to FIL source code for any fuzzy unit.

- RTC - generates code for specific chips, such as Motorola MC6805.

Fide Package

You should find in your Fide package the following items. If any item is missing or defective on arrival, contact your dealer or distributor immediately for repair or replacement of the item.

- Distribution Media
  (3 5.25" 1.2MB or 3 3.5" 1.44MB diskettes)
- Fide Hardware Key and serial number
- Fide Quick Start Guide (This Guide)
- Fide User's Manual
- Fide Reference Manual

The *Quick-Start Guide* is a set-up guide and introduction designed to quickly give you a hands-on overview of the power of Fide. It includes installation instructions, a brief tutorial on fuzzy inferencing, followed by a walk-through on using some of Fide's features.

The *Fide User's Manual* contains two parts: (1) a tutorial on fuzzy inference systems and using Fide, and (2) a comprehensive description of Fide functions available through Fide's menus.

The *Fide Reference Manual* contains technical specifications, including FIL and FCL syntax and semantics, and Run-Time Library functions. For example, it describes how you call run-time library routines from within your own Turbo C programs.

Registering Your Copy of Fide

Please take a moment to fill out and mail the registration card you received with Fide. Registering your Fide product entitles you to the following benefits:

- Automatic notification of upgrades or revisions to Fide.
- Customer support via telephone or our electronic bulletin board service called FuzzyNet.

Conventions

To help you interpret information easily, Quick-Start uses consistent visual cues and a few standard text formats. These conventions are explained as follows:

Visual Cues (typographic conventions)

This ...	Represents ...
`typewriter`	Anything that you must type exactly as it appears. For example, you type `dir fide` to see all the files in the fide main directory.
italic	Signals a new term. An explanation immediately follows the italicized term.
ALL CAPITALS	Directory names, filenames, and acronyms.
Initial Capitals	Menu items, command names, and dialog-box names and options. For example: File menu, Save command, or Course option.

Keyboard Formats (Key combinations and sequences)

This ...	Represents ...
KEY1+KEY2	Press down and hold the first key while you then press the second key. Release both keys.
KEY1, KEY2	Press and release the keys, one after the other.

Chapter 2 Installing Fide

What You Need to Run Fide

The minimum hardware and software requirements your computer system needs to run Fide successfully are shown in the following list.

- MS-DOS 3.3 or later
- MS Windows 3.0 or later
- A personal computer with an 80286 processor (or higher) and 2 megabytes (MB) or more of memory.
- A hard disk with 5 megabytes of free disk space, and at least one floppy disk drive.
- A VGA color monitor supported by Windows.
- A mouse supported by Windows. Fide requires a mouse, since it supports drawing functions that can only be utilized through mouse cursor movement, and not through equivalent keyboard commands.
- A parallel port (may be the same port used by a printer) to which the Fide hardware key (Hard Key for short) is attached.
- One of the following C compilers: Turbo C 2.0 or later, Borland C 2.0, or Microsoft C 6.0.

Running the Installation

Setting up Fide on your computer is simple. Before installing Fide on your computer, it is highly recommended that you make a backup copy of the original Fide diskette set (4 disks) and store it in safe place.

Fide runs under Microsoft Windows 3.0 or 3.1. Installation occurs under Windows control. To install Fide you must have 5 megabytes of free disk space available on your hard disk. Insert Disk 1 of your distribution media set into drive A: of your computer. Key in the following command:

```
a:> win setup
```

You will be prompted to confirm or enter in the name of the default directory to which Fide files will be copied. Unless you change the default, Fide files will be stored in the directory Fide. This prompt is shown below in Figure 2.1.

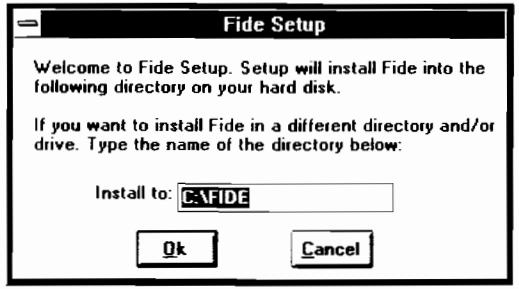

Figure 2.1 ***Fide Setup Prompt Dialog Box***

Fide Setup creates a directory called Fide (unless you have overridden the default with your own entered directory name) and several subdirectories within Fide, and then copies all Fide files into them. As the files are copied onto your hard drive, Setup routes them to the Fide subdirectories as listed below. Note that these file names may differ slightly from what is on your distribution diskettes. The latest file information is included the README file.

\FIDE
README

\FIDE\BIN
FIDEIC.EXE
FIDECPL.EXE
FIDE.EXE
FIDEMAKE.PIF
FIDE.FNT
FIDECOMP.EXE
FIDEPP.EXE

FIDEMAKE.CFG
FIDEDDB.EXE
FIDESDB.EXE
FIDEDB.EXE
FIDETR.EXE

MC6805CV.EXE
68HC05CV.EXE
68HC11CV.EXE
MC6805.CVT
MC68HC05.CVT
MC68HC11.CVT
MC6805SM.EXE
68HC05SM.EXE
68HC11SM.EXE
SUMLIS.TXT

\FIDE\LIB
FIULIBTL.LIB
FIULIBTS.LIB
FIULIBTM.LIB
FIULIBTC.LIB
FIULIBTH.LIB
FIULIBMS.LIB
FIULIBMM.LIB
FIULIBMC.LIB
FIULIBML.LIB
FIULIBMH.LIB

\FIDE\INCLUDE
FIU.H

\FIDE\EXAMPLES
PEND (subdirectory)
TRUCK (subdirectory)
ABS (subdirectory)
FIUSAMPL (subdirectory)

FANS (subdirectory)

Follow the on-screen instructions for inserting diskettes. Once all files have been installed successfully, a completion dialog box appears as shown in Figure 2.2.

Figure 2.2 *Setup Successful Dialog Box*

Fide Setup does *not* automatically modify your autoexec.bat file to include C:\FIDE\BIN in your path statement. You must perform this modification yourself. This feature is intended so that your system configuration is not changed without your knowledge.

Hardware Key

Fide will not run unless you have installed the hardware key included in your Fide package. Plug the hardware key into your computer's (IBM compatible PC) parallel port 1 (`lpt1:`). If you have a device (e.g. printer) already attached to the parallel port, detach its connector and reattach it on top of your Fide hardware key. In other words, your Fide hardware key is inserted between parallel port 1 and any pre-existing parallel device.

Chapter 3 A Primer On Fuzzy Logic

Introduction

This chapter introduces you to the basic concepts of fuzzy logic. Later chapters will introduce you the basic design of fuzzy inference systems, take you through a tour of Fide, and show how these concepts can be applied. For an extensive tour on all the aspects of Fide, see the *Fide User's Manual.*

Fuzzy logic is a powerful problem-solving methodology with applications in embedded control and information processing. Fuzzy provides a remarkably simple way to draw definite conclusions from vague, imprecise information. In an intuitive sense, it resembles human decision making. For example, refer to the following declarative assertion:

"Today's temperature is High."

If the temperature is 95 ° F, the above statement is completely true. However, if 95 ° F is High, how about 94 ° F ? 93 ° F ? ... 70 ° F ? At what degree does the statement suddenly become false? Since the temperature can change smoothly, it is hard to find a fixed value such that larger than the fixed value is High and less than the value is not High. To resolve this dilemma, we can assign partial membership degree to a temperature. A temperature is High only to a certain degree of truth, called a membership degree, or *grade*. Fuzzy logic is a mathematical formalism intended to handle these kinds of imprecise statements. It is different from traditional mathematical formalisms because fuzzy logic allows us to model the vagueness in our mental decision making process. Consequently, fuzzy logic is intuitive and, for many applications, it is easier to work with than traditional logic, which would have a difficult time processing the above statement.

Why use fuzzy logic? Its main benefits are ...

- **Lower development costs**. Engineers can describe and model solutions to a problem without needing to use complex mathematical models. This simplifies the system design process, as well as makes it easier to maintain and upgrade.

- **Better product performance**. Engineers can optimize a known solution and simplify the implementation. Fuzzy logic also offers high fault tolerance and a better tradeoff between system robustness and sensitivity.

In the design of fuzzy inference systems, there are three basic concepts that will each be explained in this chapter:

- Membership functions
- Fuzzy logic operation
- Fuzzy rules

Membership Functions

We can denote the grade of a temperature in High by assigning it a numerical "truth value". Truth values may range anywhere between 0 and 1 and are typically considered (although not always) to have meaning as suggested in the following table.

Truth Value	Meaning
0.0	Completely false
0.2	Mostly false
0.4	Somewhat false
0.6	Somewhat true
0.8	Mostly true
1.0	Completely true

In this manner we can arbitrarily assign various grades to High as a function of the temperature.

Temperature(^{O}F)	Grade in High
10	0.0
20	0.0
30	0.3
40	0.4
50	0.5
60	0.6
70	0.7
80	0.8
90	1.0

These values are illustrated on the following chart.

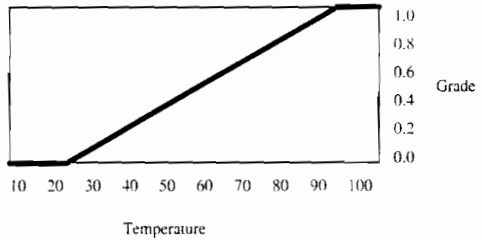

Figure 3-1 ***Membership Function of High***

The dark line in Figure 3-1 illustrates an example of how the membership function of a fuzzy concept is dependent upon a measured value, (e.g. temperature). In general, these curves are also called fuzzy sets. The fuzzy concept High is called a label.

For a temperature, such as 80^{O}F, its grade in High is 0.8, denoted as High(80)=0.8. If today's temperature is 80^{O}F, the truth value of the assertion "Today's temperature is High" is 0.8.

Fuzzy Logic Operations

The statement presented in the above section is a *simple assertion*. In addition to these, our spoken language also uses logical operators, such as "AND" or "OR" to combine simple assertions into more complex ones, called *compound assertions*. For example, let's define another simple assertion:

"The pressure is Low"

We can now create a compound assertion:

"Today's temperature is High *and* the pressure is Low"

Once we have assigned a truth value to each simple assertion, the fuzzy logic operations give us a way to precisely evaluate truth values of the compound assertions.

Before delving into the algorithms used by fuzzy logic, we first examine the results of applying Boolean logic (the backbone of digital data processing) to various combinations of truth values for these statements. This is shown in Table 3-1. We then compare them to results we'd get from applying fuzzy logic, also shown in Table 3-1. Note that both Boolean logic and fuzzy logic produce the same results when the statements are limited to 1 and 0. However, for values other than 1 or 0, fuzzy logic produces meaningful results while Boolean logic is simply not defined to handle such cases.

Table 3-1 ***Fuzzy Logic Operations***

		Boolean Logic		Fuzzy Logic	
A	B	A And B	A Or B	A And B	A Or B
1	1	1	1	1	1
1	0	0	1	0	1
0	1	0	1	0	1
0	0	0	0	0	0
0.7	1	?	?	0.7	1
0.7	0.8	?	?	0.7	0.8
0.7	0	?	?	0	0.7
0.5	0.5	?	?	0.5	0.5

Where A=Today's temperature is High, B=The pressure is Low

There is a special mathematical function used by fuzzy logic to evaluate truth values for statements containing the AND or OR operators. For two arbitrary statements X and Y (which can be either simple or compound), the functions are ...

compound operation on statements X , Y	mathematical function to evaluate truth
X AND Y	minimum
X OR Y	maximum

For an example, let the truth value of X be 0.7 and that of Y be 0. Then

minimum = minimum (0.7, 0.0) = 0.0
maximum = maximum (0.7, 0.0) = 0.7

That is, the truth value of 'X And Y' is 0, that of 'X Or Y' is 0.7.

Fuzzy Inference Rules

Inference rules describe dependent relationships among variables. If, for example, in a temperature control process, we have two fans - one is for hot air and the other for cool air - to control the temperature, then the variables are:

temperature = current temperature;
hot_fan_spd = the speed of the hot air fan;
cool_fan_spd = the speed of the cool air fan;

The control rules may be

if temperature is high then hot_fan_spd is zero
if temperature is high then cool_fan_spd is high
if temperature is low then hot_fan_spd is high
if temperature is low then cool_fan_spd is zero

Note that a *rule* has the following skeletal format:

IF <*condition*> THEN <*consequence*>

The *condition* part of a rule could be either a simple or compound statement and the *consequence* part is typically a simple statement.

There are many fuzzy inference methods to process/interpret the rules. For details on all the methods available in Fide see the *Fide User's Manual*. The next chapter in this Quick-Start manual will describe one of the most popular "toy" problems in system control and how fuzzy logic can be used to design a fuzzy inference controller that solves the problem.

Chapter 4 Fuzzy Inference Design

Introduction

This chapter gives an example of how fuzzy inference can be applied toward controlling a simple, yet popular system: an inverted pendulum-cart balancer. We will go through all the calculations only once so that you can obtain a birds-eye view of what Fide does automatically. In the next chapter, you will have the opportunity to run Fide and see the pendulum-cart balancer in operation.

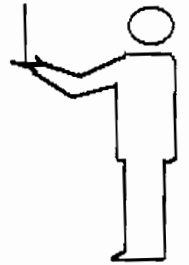

Figure 4-1 ***An Acrobatic Master***

What is a pendulum-cart balancer? An acrobatic master can place a broomstick on his finger and keep it balanced by moving his hand beneath it (see Figure 4-1). The acrobatic master can train students by relating relevant knowledge to them in the form of rules. The rules may be:

IF the stick is declining left THEN move your hand to the left.
IF the stick is declining right THEN move your hand to the right.
IF the stick is declining slowly THEN move your hand slowly.
IF the stick is declining rapidly THEN move your hand rapidly.

We shall now see how the principles alluded to in the previous chapter give us a way to apply these rules in order to model the acrobatic master's performance. This would seem straightforward because the rules taught by the master are

simple. Unfortunately, such simple rules are difficult to implement in computer software because they are neither arithmetic (numerical), nor symbolic, nor can they be realized using standard logic. Fuzzy logic, on the other hand, provides a systematic interpretation of the acrobatic master's rules. The required steps for building such a model are ...

- define input and output variables
- create inferencing rules
- create membership functions
- select a defuzzificaton method

These steps are discussed in the following sections.

Defining Variables

Suppose an engineer is trying to establish an inverted pendulum simulating the master's performance: a stick supported by a cart with a joint. The stick rotates around the joint (Figure 4-2). The input and output signals of the cart-pendulum system are shown in Figure 4-3. A controllable motor moves the cart in either direction according to the control signal sent to it. The control signal is represented by the variable *volt*, the voltage of the signal sent to the motor. The cart moves to the left if *volt* is positive. Furthermore, a sensor measures two aspects of the pendulum: *theta*, the declination angle of the stick, and its time derivative *omega*, the angular velocity. (The decline angle is defined to be positive when the stick is declining to the left).

Note that since the *fuzzy inference unit* (FIU) we will design is intended to control this cart-pendulum system, then the system's output becomes the FIU's input, and vice-versa.

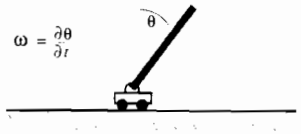

Figure 4-2 ***Inverted Pendulum System***

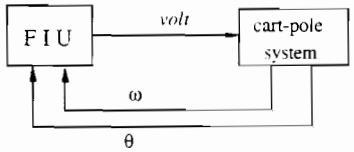

Figure 4-3 ***Inverted Pendulum Control System***

Creating Inference Rules

The rules supplied by the acrobatic master need to be modified so that they incorporate the input and output variables. For *theta* and *volt*, the rules could be reworded to the following:

IF theta is positive THEN volt is positive.
IF theta is negative THEN volt is negative.
IF theta is large (in magnitude), THEN volt is large (in magnitude).
IF theta is medium (in magnitude), THEN volt is medium (in magnitude).
IF theta is small (in magnitude), THEN volt is small (in magnitude).

A similar set of rules could also be written for variables *omega* and *volt*. However, in order to deal with the coupling effects between the two input variables, we shall incorporate all three variables in our rules. An example would be ...

IF theta is a large negative value AND omega is a small positive value
THEN volt is a large negative value.

In order to make the rules more concise and readable, the following notation is used:

Label	Description
NL	negative large
NM	negative medium
NS	negative small
PL	positive large
PM	positive medium
PS	positive small

The rules can now be written to resemble the following examples ...

If theta is NL and omega is PS then volt is NL.
If theta is NS and omega is PL then volt is PM.
If theta is PM and omega is NL then volt is NS.

All possible combinations of *theta* and *omega* with their consequent values for *volt* can be summarized in Table 4-1.

Table 4-1 ***A Set of Rules***

theta	*omega*					
	NL	*NM*	*NS*	*PS*	*PM*	*PL*
NL	*NL*	*NL*	*NL*	*NL*	*NM*	*NS*
NM	*NL*	*NL*	*NL*	*NM*	*NS*	*PS*
NS	*NL*	*NL*	*NM*	*NS*	*PS*	*PM*
PS	*NM*	*NS*	*PS*	*PM*	*PL*	*PL*
PM	*NS*	*PS*	*PM*	*PL*	*PL*	*PL*
PL	*PS*	*PM*	*PL*	*PL*	*PL*	*PL*

Defining Membership Functions

Note that the above rules are fuzzy because the labels *large*, *medium* and *small* are not well-defined in the sense of precise physics or mathematics. Consequently, we need to create a membership function for each of the six labels NL, NM, NS, PL, PM and PS for variables *theta*, *omega*, and *volt*.

The first step is to define the upper and lower limits of each variable. In the next step, we can assume the six membership functions for *theta* are the same as the six functions for *omega*. We shall also use the six arbitrarily defined (but intuitively appealing) membership functions as shown in Figure 4-4. The dashed curve labeled NL is the NL membership function; likewise for all the other curves.

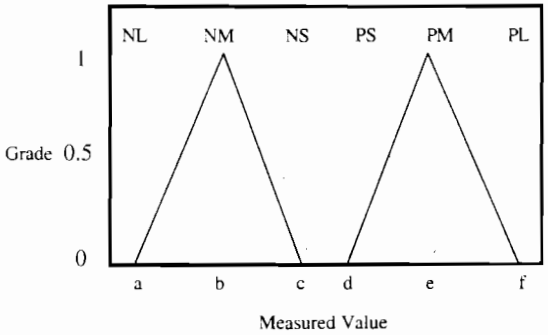

Figure 4-4 ***Membership Functions***

For these membership functions, we need to specify the grade at critical points a,b,...,f. For example, membership function NM may be specified as:

NM (@a,0, @b,1 @c,0)

That is, NM has grade 0 at point a, grade 1 at point b, and grade 0 at point c. The grades at other points are linearly interpolated.

Using Defuzzification

Recall that we desire from all this effort a single value of *volt*, the voltage to be applied to the motor in the pendulum-cart system. The process of reducing numerous claims by fuzzy rules into one "crisp" value is called *defuzzification*. There are many defuzzification methods (see *Fide Reference Manual*). A commonly used one is the centroid method (also called COG - center of gravity - method).

For example, suppose the output truth values for the labels of volt are:

Label	Truth Value
NL	0
NM	0
NS	0.5
PS	1
PM	1
PL	0.5

By using the centroid method, the crisp output value is 164 (see Figure 4-5).

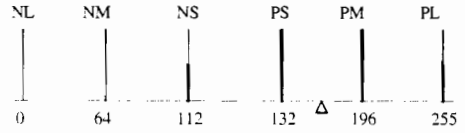

Figure 4-5 ***Fuzzy Output and Defuzzification***

Chapter 5 A Brief Look At Fide

Introduction

This chapter introduces you to some of the major aspects of Fide by presenting a step-by-step interactive tour through Fide using a pendulum-cart system. A full description of Fide is provided in the User's Guide.

To speed you along, all the software for the pendulum-cart system is provided. You will learn how to modify various portions of the program using Fide and observe the results. In particular, you will ...

- See what FIL (codes in Fuzzy Inference Languages) looks like and make changes through the editor.
- Set specific input data and trace the details of the inference process, step-by-step.
- View a graphic representation of the system's global input-output response transfer function.

As shown in Figure 5-1, the pendulum-cart problem we will use is actually composed of two tasks: to balance the pendulum (i.e. get theta = 0) and to center the cart on its track (i.e. get x = midpoint location).

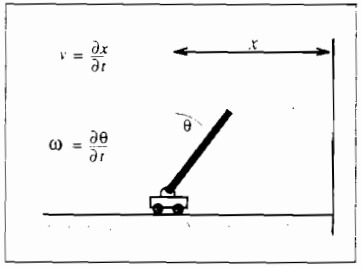

Figure 5-1 ***A Pendulum-Cart System***

To create this or any other Fide fuzzy inference system, a base source file is needed. A base source file is the fundamental building block. It contains the definitions of all variables, rules and membership functions for one inference unit. Each base source file is called a Fide Inference Unit (FIU). If you want to use several different inference units within a system, you'll need to make one FIU for each.

The pendulum-cart example uses two FIU files, named PNDRULE1.FIL and PNDRULE2.FIL:

PNDRULE1 balances the pendulum, and
PNDRULE2 returns the cart to the center of its track.

Often the user wants to modify data before entering an FIU. For example, both the measured angle of the pendulum and its angular velocity need to be normalized to a value in certain ranges. Other mathematical operations may be desired as well. If an input variable of an FIU is to be modified, the mathematical operation is specified in a Fide Operating Unit (FOU).

The pendulum-cart example uses four FOU files, named OP1, OP2, OP3, and OP4. As shown in Figure 5-2, we see that

OP1 normalizes and scales cart position x.
OP2 normalizes and scales cart movement speed dx.
OP3 combines pendulum angle t and the output of FIU1.
OP4 normalizes and scales pendulum angle velocity dt.

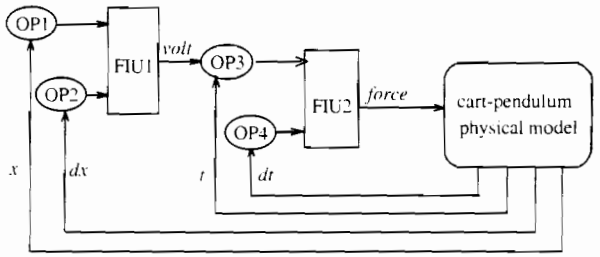

Figure 5-2 ***Inverted Pendulum Control System***

Fide runs under Microsoft Windows 3.x. It is assumed that the reader is familiar with the basics of MS Windows operations. If not, please read your Microsoft Windows User's Guide before proceeding. In this chapter, sub-menu commands will be written as : *Menu1/Menu2/Menu3* ...

where *Menu1* is the top level Menu item
Menu2 is a menu item under Menu1
Menu3 is a menu item under Menu2, and so on.

Instructions for you to interact with the computer will be printed in `Monospaced type`.

Starting Fide

After Fide has been successfully installed, you can start Fide from either DOS or Windows as follows:

From DOS

```
cd c:\fide\bin
win fide
```

From Windows

If you do not have a Fide icon (this should be created automatically for you when install Fide) on your desktop, you should first create one in the manner described in the Windows User's Guide. Double-click on the Fide icon.

Either way, you will get the base window of Fide, as shown in Figure 5-3.

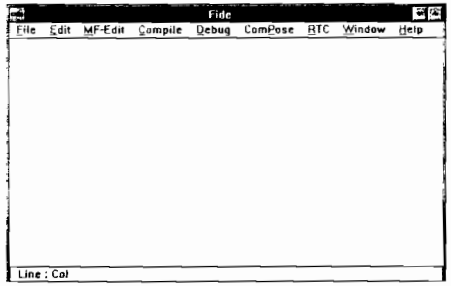

Figure 5-3 ***Fide Window***

FIU Files

New FIU files can be created from scratch; however, for the purpose of this Quick-Start, load in the pre-existing FIU file named PNDRULE2.FIL as follows:

1. Select File/Open from the Fide window menu. A file selection dialog box appears.
2. Select by double-clicking the file named "PNDRULE1.FIL" from the subdirectory Fide\examples\pend.

PNDRULE1.FIL is displayed in a window that occupies about half the base window. It is useful to know at this point that multiple files can be opened; however, only one window per file is allowed. In other words, you may not have multiple instances of any file.

PNDRULE1.FIL contains, among other things, definitions of input variables and their membership functions. Figure 5-4 shows these specifications for input variable "pend_angle".

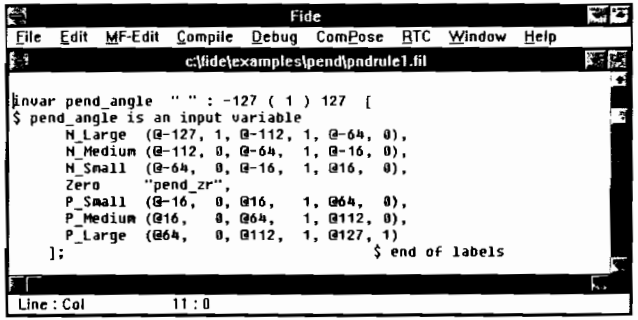

Figure 5-4 ***Specification of an Input Variable***

Further down in this FIU file is the collection of rules as described in the previous chapter. Some of those rules are shown in Figure 5-5.

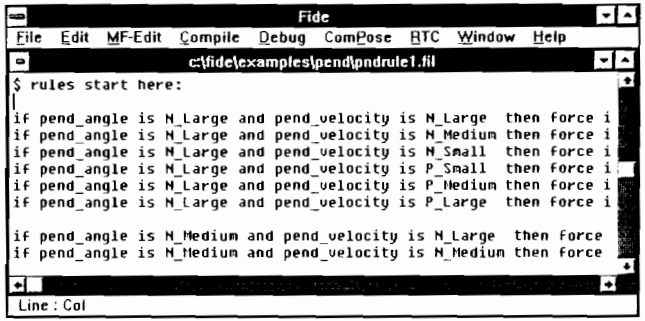

Figure 5-5 *Fuzzy Rules*

All FIU (and FOU or FEU files, if any) must be compiled before they can be used numerically in any fuzzy program. Fide will attempt to compile the file whose window is active. Upon completion of a compilation, a message window will appear immediately below the source file window. It will contain a list of error messages, if any, from which the user can correct detected problems.

To see the compiler's error detection capabilities in action, do the following:

1. Add the line "hello" just above the line saying "invar pend_angle..." as shown in Figure 5-6.

2. Save the file by selecting File/Save from your Fide window menu.

3. Compile the saved file by selecting Compile/Compile Source from your Fide window menu.

```
hello
invar pendulum_angle  " " : -127 ( 1 ) 127 [
```

Figure 5-6 *Wrong Syntax*

As shown in Figure 5-7, the Messages window will complain that there is a "function" which does not have the expected number of parameters. To see what function it is talking about, do the following:

Double-click on the first error message in the Message window.

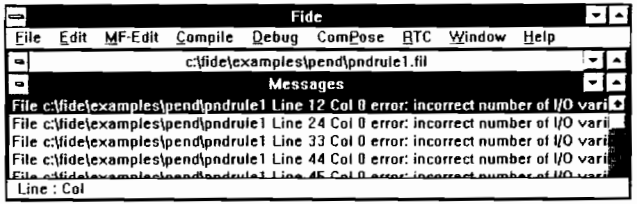

Figure 5-7 *Error Messages*

Note that the first character of "HELLO" has been highlighted in the PNDRULE1.FIL file window. To correct this intentional coding error, do the following:

1. Delete the line containing "hello" in the PNDRULE1 file window.

2. Save the file by selecting File/Save from your Fide window menu.

3. Compile the saved file by selecting Compile/Compile Source from your Fide window menu.

If you have performed this correctly, you should now get a small window declaring the compilation was successful. To remove the notice and the FIU file, do the following:

1. Click on the OK button.

2. Select the Close option from the File menu Figure 5-8.

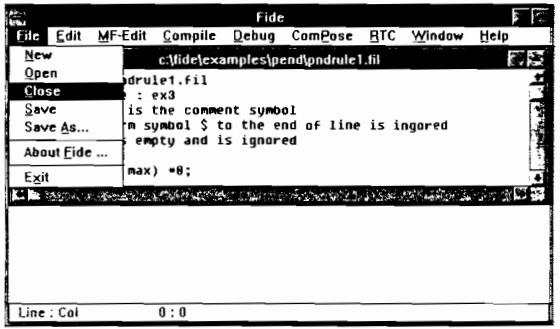

Figure 5-8 *Close a File*

FOU Files

New FOU files can be created from scratch: however, for the purpose of this Quick Start, load in the pre-existing FOU file named OP1.FIL as follows:

1. Select File/Open from the Fide window menu. A file selection dialog box appears.
2. Select by double-clicking the file named "OP1.FIL" from the subdirectory Fide\examples\pend.

OP1.FIL is displayed in a window that occupies about half the base window. It contains a small mathematical formula designed to normalize a single input into a single output. See Figure 5-9. The definitions of variables *x* and *y* are local to this FOU only, thus allowing the use of *x* and *y* in other FOUs without any problem of erroneous interaction. In fact, variable *y* is found in all four FOU files used in the pendulum-cart program.


```
fou ;
> x;
< y;
calculate y=x*20+127
end
```

Figure 5-9 *An FOU File*

Each FOU file needs to be compiled before it can be used. To close the FOU file, do the following:

Select the Close option from the File menu to close the OP1.FIL file window. The location is the same as shown in Figure 5-8.

FEU Files

The user may want to have the Fide inference system interact with non-fuzzy Turbo C program modules, such as callable graphics routines for displaying the data and callable models of physical systems (e.g. a cart-pendulum model). If this is the case, the user will need to identify for Fide the C routines to be called, as well as their input and output parameters. These specifications are created in files called Fide Execution Units (FEU). One FEU file is created for each location in the fuzzy system's dataflow where a call to an external object C module is made.

New FEU files can be created from scratch: however, for the purpose of this Quick Start, load in the pre-existing FEU file named PDEMO.FIL as follows:

1. Select File/Open from the Fide window menu. A file selection dialog box appears.
2. Select by double-clicking the file named "PDEMO.FIL" from the subdirectory Fide\examples\pend.

PDEMO.FIL is displayed in a window that occupies about half the base window. It contains input and output definitions to the external C module "pdemo" as well as a call to the module. See Figure 5-10. The "pdemo" external module models the cart-pendulum physical system. It inputs the control force (specified by parameter *f*) and output the pendulum angle, angular velocity, cart position and cart velocity (specified by parameters *t*, *dt*, *x*, *dx*, respectively).

```
feu "invpend1";
>f;
< x;
<dx;
<t;
<dt;
call  pdemo
end
```

Figure 5-10 ***pdemo Statements***

Each FEU file needs to be compiled before it can be used. To close the FEU file, do the following:

Select the Close option from the File menu to close the PDEMO.FIL file window. The location is the same as shown in Figure 5-8.

Debug Methods

Fide offers three different ways to examine the behavior of an FIU:

- Tracer - tracks specific FIU behavior back to source rules.
- Analyzer - graphically reveals the transfer function of an FIU.
- Simulator - displays the dynamic input-output relationship over time.

This section of Quick Start will show how to display the transfer function of fuzzy rule PNDRULE1.FIL and to notice how it changes as one of the rule's

membership functions undergoes modification. To do this, you must first make sure the FIU compiles correctly, then activate the Analyzer.
Do the following:

1. Select File/Open from the Fide window menu. A file selection dialog box appears.
2. Select by double-clicking the file named "PNDRULE1.FIL" from the subdirectory Fide\examples\pend.
3. Compile the file by selecting Compile/Compile Source from your Fide window menu. If you get the message that compilation was successful, then proceed. If not, correct the problem before proceeding with this exercise.
4. Select Debug/Run_Debugger in the Fide window menu. This activates the Debugger application, which has its own menu bar for selecting Tracer, Analyzer or Simulator.
5. Select Tools/Analyzer in the Debugger window menu. This invokes a Select and Edit dialog box. Select all three variables by clicking on them.
6. Press the Execute button and select the "course" precision radio button. Then press the OK button. During execution, the FIU is tested with many pairs of input values in order to build the transfer function.
7. When the hourglass becomes a mouse pointer, press the Display button. This brings up the Analyzer application window.
8. Select Views/Surface from the Analyzer window menu. This causes the Analyzer to draw a 3-D image of the rule's transfer function, as shown in Figure 5-11.
9. Click and drag on the small red arrows by the base of the image. As you do so, the two input values as well as the output value are displayed on the right.

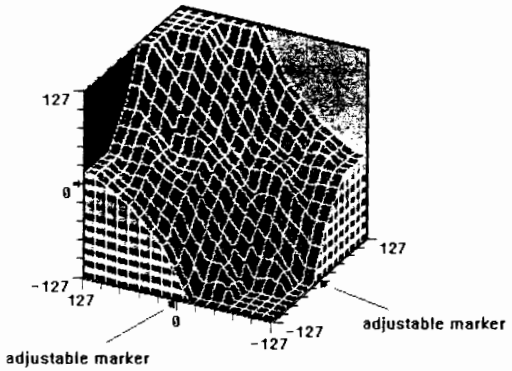

Figure 5-11 *A Control Surface*

10. Press the Exit button to remove the 3-D draw window. Remove the Analyzer window by double-clicking on the upper left hand corner of window. Remove the Debugger window in the same way.

You are now back in the Fide window. The next step is to modify one or more of the membership functions to the input variable denoting pendulum angle and then redraw the 3-D transfer function image to see how it has changed. To modify the shape of a membership function, do the following:

11. Select MF-Edit/Edit in the Fide window menu. This invokes the Membership Function Editor window.

12. Highlight the variable labeled "pend_angle" by clicking on it and then press the Input Label button. This opens up the Membership Function Editor window.

13. Select different radio buttons on the right and notice how the corresponding membership function curve is highlighted. See Figure 5-12.

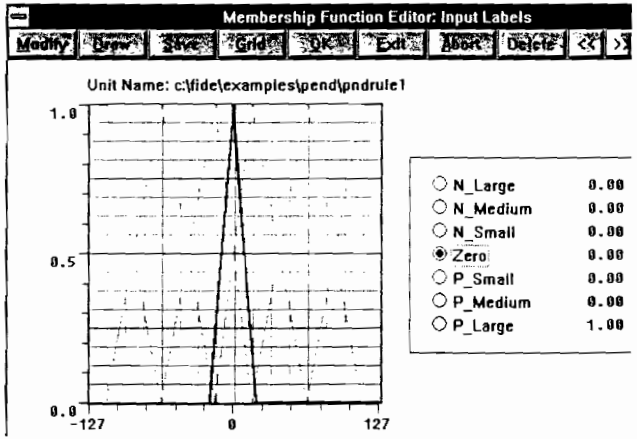

Figure 5-12 *Membership Function Editor*

14. Slowly move the mouse cursor horizontally across the grid and watch the corresponding truth values change in a box to the right.

15. Press the Modify button. This brings up the Label Dialog window.

16. Select the label "Zero" by highlighting it and then click on Draw.

17. Move the pointer on the graph and click where the points of the label should be located. Use the delete button to erase any point you are re-defining and do not like (see Figure 5-13).

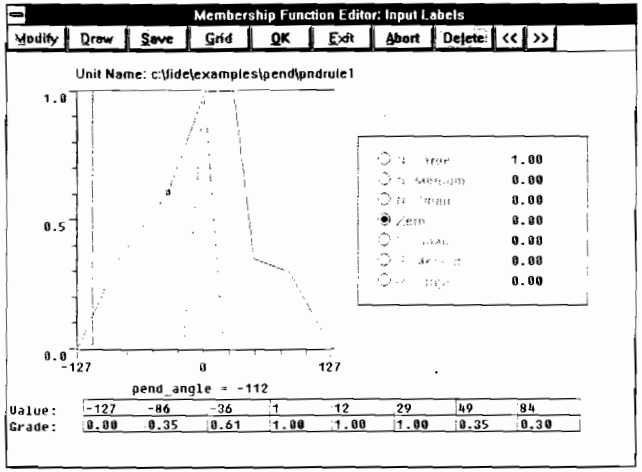

Figure 5-13 ***Editing Membership Function***

18. Press the Save button on the Membership Function Editor window. This saves the functions for the FIU.

19. Recompile the FIU.

At this point you can get Fide to redraw the 3-D image of the rule's transfer function. Do this by re-executing steps 4-9. Remove the windows by executing step 10.

Composer

After all FIU, FOU and FEU files have been compiled, they are combined into a whole system by the Fide tool called Composer. In order to combine the files,

Composer needs to know how the units are connected, that is to say, it needs to know what data is being transferred between what units. This information is collected in a pair of files, one containing the information in text format (the *.LNK files) and the other containing the same data in graphic format (the *.GNK files). For the pendulum-cart program, these two files have already been created. To see the graphic representation, do the following:

1. Select Composer/Switch_to_Composer in the Fide window menu. You are now running the Composer module.

2. Select File/Open_As_Graphics in the Composer window menu.

3. Select by double-clicking the file named "PEND1SYS.GNK" from the subdirectory Fide\examples\pend.

The drawing on your screen should match Figure 5-14.

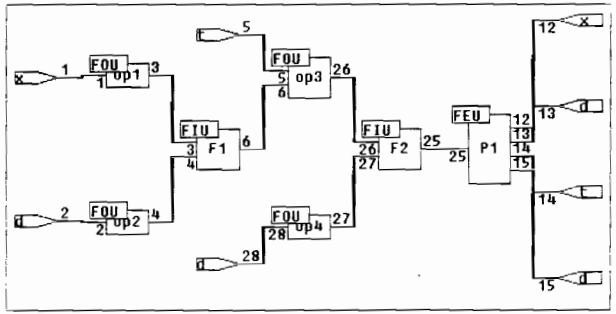

Figure 5-14 ***Composer Graphic Representation***

Note how the FEU is used to model the physical cart-pendulum system, inputting only one signal (the control force) and outputting the four measurable system state variables t, dt, x, dx. The user can inspect what variables are used

as input and output to any unit. To see what variable is input to the FEU, do the following:

1. Click on the magnifying glass icon in the menu bar of the Composer window. This turns your cursor into an inspector.
2. Double-click on the box labeled "F2". This produces a pop-up window showing its output to be "force". This is input to the FEU.
3. To close the pop-up window, click on the OK button.

With Composer, the user has two ways to observe the behavior of the whole system over time:

- **Make For Dataflow** command: links all the units together, so during run time, it presents the output data in numerical format of all FIU, FEU and FOU units.
- **Make for Simulation** command: links all the units together, so it presents a response curve of selectable system input-output variables after the simulation has run its course.

To execute either method, your computer must have Borland's Turbo C compiler. To see a behavioral response curve, do the following:

1. Select Make/Make_For_Simulation in the Composer window menu. This creates an executable (*.EXE) file.
2. Select Run/Set_Value in the Composer window menu. Set initial values for the input variables: x=10, dx=0, t=1, dt=0. To select a variable for initialization, double-click on it.
3. Select Run/Run in the Composer window menu. This runs the executable file thereby generating the desired response data.

The system will execute the default number of 1000 steps of simulation. While it is running, the mouse cursor is converted into an hourglass indicating that Composer is busy and no other work may be performed during that time.

4. Select Debug/Display_Simulation in the Composer window menu.

This opens a window entitled "Select Variables". In this window, the variables of the selected system are listed and any number (up to 7) of system input and output variables may be selected for plotting.

5. Select the desired variables for plotting by double-clicking on them.
6. Close the pop-up window by clicking the Display button. A display of a graphic plot appears, representing the temporal changes in the selected variables.
7. Move the mouse into the graph region. As you move the mouse, the simulation step number and variable's value are displayed. Clicking on the various radio buttons selects the variable.

To exit Composer, do the following:

8. Select Exit from File menu.

Chapter 6 Customer Service

Technical Support Services

Aptronix is committed to providing the highest level of customer support possible. There are 3 means by which users of Fide can contact Aptronix for technical support. They are:

FuzzyNet

Aptronix maintains an internal BBS (Bulletin Board System), called FuzzyNet, where interested parties can download sample files, technical updates, and other information on Fide related issues through use of a modem. Users can also use the BBS to upload source files or other pertinent information to Aptronix for special attention. FuzzyNet also supports E-Mail messages for on-line communications with Fide users. No special setup, other than a modem and basic communications software, is required.

FAX

Users can FAX in a technical request to Aptronix with a return FAX number or phone number, and Aptronix will respond to the request either with a suggested solution or a time frame for a solution.

Voice

Aptronix can be contacted directly via telephone between the hours of 9:00am and 5:00pm Pacific Standard Time. If the question is about Fide, please have your product serial number available and call from a phone near the computer on which you are running Fide.

APTRONIX CONTACT NUMBERS

•	Phone	408-428-1888	hours: 9am - 5pm PST
•	Fax	408-428-1884	hours: anytime
•	FuzzyNet	408-428-1883	hours: anytime

Custom Engineering Services

The Aptronix Engineering Group provides immediate assistance with all aspects of fuzzy logic system development. The Group's expertise ranges from modular design and implementation all the way through to complete systems. Aptronix engineers have years of experience in fuzzy logic applications -- expertise that can help customers create better products faster. For more information, call or write to:

Aptronix, Inc.
2150 North First Street, Suite 300
San Jose, California 95131